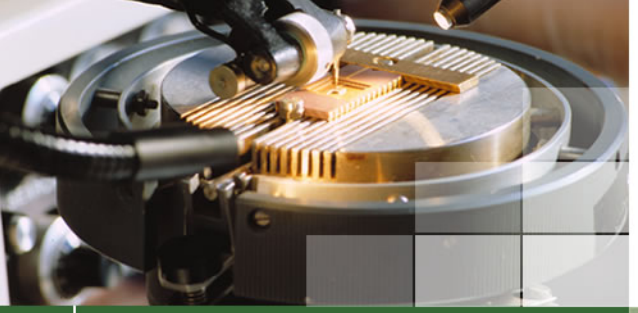


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Planning and Managing Software Projects 2013-14

Class 8

Estimation

Emanuele Della Valle

http://emanueledellavalle.org

Credits

- This slides are largely based on Prof. John Musser class notes on “Principles of Software Project Management”
- Original slides are available at http://www.projectreference.com/
- Reuse and republish permission was granted

Agenda

- Last Class
  - Review Classes 5 and 6
  - Work Breakdown Structures (WBS)
- Today
  - Brief Class 7 review
  - Estimation

Estimation

- "Predictions are hard, especially about the future"

  - Yogi Berra*

* http://en.wikipedia.org/wiki/Yogi_Berra

Planning, Estimating, Scheduling

- Plan:
  - Identify activities. No specific start and end dates.
- Estimating:
  - Determining the size & duration of activities.
- Schedule:
  - Adds specific start and end dates, relationships, and resources.

WBS

- Hierarchical list of project's work activities
- WBS becomes input to many things
  - Network scheduling
  - Costing
  - Risk analysis
  - Organizational structure
  - Control
  - Measurement
- What hurts most is what's missing

WBS Types and Formats

- Types:
  - Process
  - Product
  - Hybrid
  - Less frequently used ones: organizational and geographic
- Formats:
  - Outline
  - graphical (similar to an organizational chart)

WBS Levels

- Up to six
- Usually 3-6
- Level 1-3: Managerial
  - Level 1: authorizations
  - Level 2: budgets
  - Level 3: schedules
- Level 4-6: Technical

WBS Guidelines

- List of Activities, not Things
- Describe activities using “bullet language”
  - Meaningful but terse labels
- All WBS paths do not have to go to the same level
- Do not plan more detail than you can manage

WBS Techniques

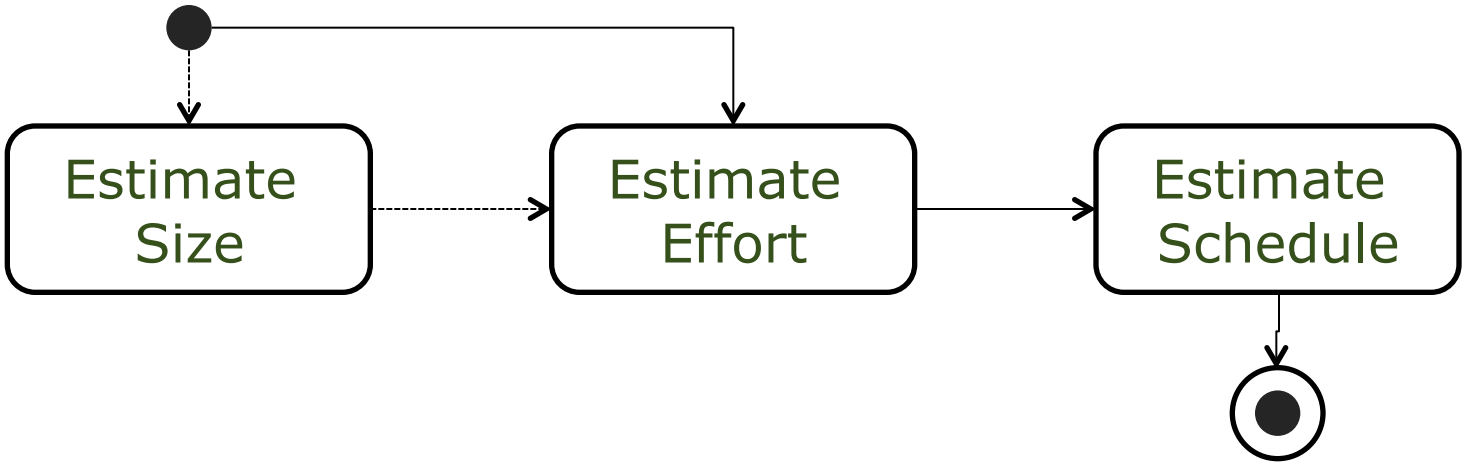
- Top-Down
- Bottom-Up
- Analogy
- Brainstorming
- Rolling Wave

Homework – 2: WBS

- Create a WBS for your project
- Guidelines
  - Do it at managerial level (see slide 14)
  - Choose the type of WBS among process, product or hybrid approach
  - Follow the standard hierarchical numbering scheme for WBS structures
  - Use outline format
- Submission
  - Use the tool you prefer between Notepad/Word/Excel
  - Add homework-2 to the appropriate folder in the dropbox folder of your project

Introduction

- Very difficult to do, but needed often
- Created, used or refined during
  - Strategic planning
  - Feasibility study and/or SOW
  - Proposals
  - Vendor and sub-contractor evaluation
  - Project planning (iteratively)
- Basic process



Introduction (cont.)

- Remember, an “exact estimate” is an oxymoron
- Estimate how long will it take you to get home from class this afternoon
  - On what basis did you do that?
  - Experience right?
  - Likely as an “average” probability
  - For most software projects there is no such ‘average’
- Most software estimations are off by 25-100%

Measuring Size

- Lines of Code (LOC)
- Function points
- Feature points or object points
- Other possible
  - Number of bubbles on a DFD
  - Number of of ERD entities
  - Number of processes on a structure chart
- LOC and function points most common
  - (of the algorithmic approaches)
- **Majority of projects use none of the above**

LOC

- Advantages
  - Commonly understood metric
  - Permits specific comparison
  - Actuals easily measured
- Disadvantages
  - Difficult to estimate early in cycle
  - Counts vary by language
  - Many costs not considered (ex: requirements)
  - Programmers may be rewarded based on this
    - Can use: # defects/# LOC
  - Code generators produce excess code

LOC Estimate Issues

- How do you know how many in advance?
- What about different languages?
- What about programmer style?
- Stat: avg. programmer productivity: 3,000 LOC/yr
- LOC Estimate are more effective after requirements (or have to be after)

Function Points

- Software size measured by number & complexity of functions it performs
- More methodical than LOC counts
- House analogy
  - House's Square Feet ~= Software LOC
  - # Bedrooms & Baths ~= Function points
  - Former is size only, latter is size & function
- Three basic steps

Function Point Process

1. Count # of biz functions per category
   - Categories: outputs, inputs, db inquiries, files or data structures, and interfaces
2. Establish Complexity Factor for each and apply
   - Simple, Average, Complex
   - Set a weighting multiplier for each (0->15)
   - This results in the "unadjusted function-point total"
3. Compute an "influence multiplier" and apply
   - It ranges from 0.65 to 1.35; is based on 14 factors
4. Results in "function point total"
   - This can be used in comparative estimates

- For a tutorial: http://www.devdaily.com/FunctionPoints/

Effort Estimation

- Now that you know the “size”, determine the “effort” needed to build it
- Various models: empirical, mathematical, subjective
- Expressed in units of duration
  - Person-Months (or Man-Months, less politically correct)

Effort Estimation

- McConnell shows schedule tables for conversion of size to effort
  - Again, not seen in ‘average’ projects
- Often the size and effort estimation steps are combined (not that this is recommended, but is what often is done)
- “Commitment-Based” Scheduling is what is often done
  - Ask developer to ‘commit’ to an estimate (his or her own)

Target vs. Committed Dates

- Target:
  - Proposed by business or marketing
  - Do not commit to this too soon!
- Committed:
  - Team agrees to this
  - After you've developed a schedule

Methodologies

- Top-down
- Bottom-up
- Analogy
- Expert Judgment
- Wideband Delphi
- Priced to Win

Can be also used to directly estimate effort

- Parametric Methods
  - Requires size estimation
  - Based on formulas and equations

Top-down Estimation

- Based on overall characteristics of project
  - Some of the others can be “types” of top-down (Analogy, Expert Judgment, and Algorithmic methods)
- Advantages
  - Easy to calculate
  - Effective early on (like initial cost estimates)
- Disadvantages
  - Some models are questionable or may not fit
  - Less accurate because it doesn’t look at details

Bottom-up Estimation

- Create WBS
- Add from the bottom-up
- Advantages
  - Works well if activities well understood
- Disadvantages
  - Specific activities not always known
  - More time consuming

Expert Judgment

- Use somebody who has recent experience on a similar project
- You get a “guesstimate”
- Accuracy depends on their ‘real’ expertise
- Comparable application(s) must be accurately chosen
  - Systematic
- Can use a weighted-average of opinions

Estimation by Analogy

- Use past project
  - Must be sufficiently similar (technology, type, organization)
  - Find comparable attributes (ex: # of inputs/outputs)
  - Can create a function
- Advantages
  - Based on actual historical data
- Disadvantages
  - Difficulty ‘matching’ project types
  - Prior data may have been mis-measured
  - How to measure differences – no two exactly same

Wideband Delphi

- Group consensus approach
- Rand corp. (http://www.rand.org/) used original Delphi approach to predict future technologies
- Present experts with a problem and response form
- Conduct group discussion, collect anonymous opinions, then feedback
- Conduct another discussion & iterate until consensus
- Advantages
  - Easy, inexpensive, utilizes expertise of several people
  - Does not require historical data
- Disadvantages
  - Difficult to repeat
  - May fail to reach consensus, reach wrong one, or all may have same bias

Priced to Win

- Just follow other estimates
- Save on doing full estimate
- Needs information on other estimates (or prices)
- Purchaser must closely watch trade-offs
- Price to lose?

Parametric Method, e.g. COCOMO

- The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm
- The model uses a basic regression formula with parameters that are derived from historical project data and current project characteristics.
- COCOMO applies to three classes of software projects:
  - **Organic projects** - "small" teams with "good" experience working with "less than rigid" requirements
  - **Semi-detached projects** - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
  - **Embedded projects** - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.(hardware, software, operational, ...)

Source: http://en.wikipedia.org/wiki/COCOMO

Parametric Method, e.g. COCOMO (cont.)

- The basic COCOMO equations take the form
  - **Effort Applied (E)** = a (KLOC)^b
  - **Development Time (D)** = c(Effort Applied)^d
  - **People required (P)** = Effort Applied / Development Time
- where, **KLOC** is the estimated number of delivered lines (expressed in thousands) of code for project. The coefficients *a*, *b*, *c* and *d* are

Software project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Source: http://en.wikipedia.org/wiki/COCOMO

Parametric Method Issues

- Remember: most projects you'll run into don't use these
- Which is 'normal', so don't be surprised
  - Or come-in to new job and say "Hey, let's use COCOMO"
- These are more effective on large projects
  - Where a past historical base exists
- Primary issue for most projects are
  - Lack of similar projects
    - Thus lack of comparable data

Issues

- Quality estimations needed early but information is limited
- Precise estimation data available at end but not needed
  - Or is it? What about the next project?
- Best estimates are based on past experience
- Politics of estimation:
  - You may anticipate a “cut” by upper management
- For many software projects there is little or none
  - Technologies change
  - Historical data unavailable
  - Wide variance in project experiences/types
  - Subjective nature of software estimation

Code Reuse & Estimation

- Does not come for free
- Code types: New, Modified, Reused
- If code is more than 50% modified, it's "new"
- Reuse factors have wide range
  - Reused code takes 30% effort of new
  - Modified is 60% of new
- Integration effort with reused code almost as expensive as with new code

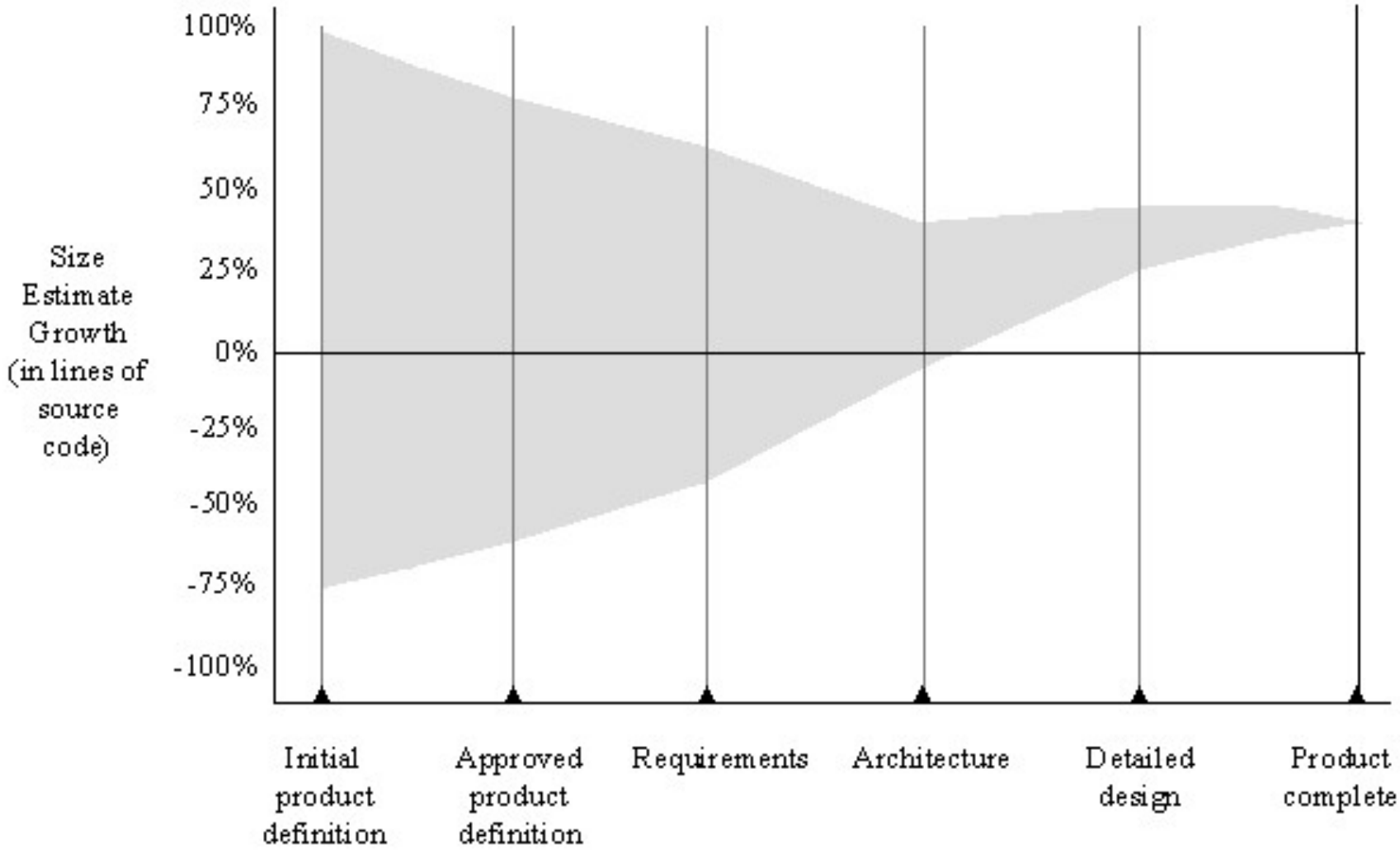
Size and Estimation Variance

- Small projects (10-99 FPs), variance of 7% from post-requirements estimates
- Medium (100-999 FPs), 22% variance
- Large (1000-9999 FPs) 38% variance
- Very large (> 10K FPs) 51% variance

Over and Under Estimation

- Over estimation issues
  - The project will not be funded
    - Conservative estimates guaranteeing 100% success may mean funding probability of zero.
  - Parkinson's Law: Work expands to take the time allowed
  - Danger of feature and scope creep
  - Be aware of "double-padding": team member + manager
- Under estimation issues
  - Quality issues (short changing key phases like testing)
  - Inability to meet deadlines
  - Morale and other team motivation issues

Cone of Uncertainty



Estimation Guidelines

- Estimate iteratively!
  - Process of gradual refinement
  - Make your best estimates at each planning stage
  - Refine estimates and adjust plans iteratively
  - Plans and decisions can be refined in response
  - Balance: too many revisions vs. too few

Know Your Deadlines

- Are they ‘Real Deadlines’?
  - Tied to an external event
  - Have to be met for project to be a success
  - Ex: end of financial year, contractual deadline, Y2K
- Or ‘Artificial Deadlines’?
  - Set by arbitrary authority
  - May have some flexibility (if pushed)

"Presentation"

- How you present the estimation can have huge impact
- Techniques
  - Plus-or-minus qualifiers
    - 6 months +/-1 month
  - Ranges
    - 6-8 months
  - Risk Quantification
    - +/- with added information
    - +1 month of new tools not working as expected
    - -2 weeks for less delay in hiring new developers
  - Cases
    - Best / Planned / Current / Worst cases
  - Coarse Dates
    - Q3 2009
  - Confidence Factors
    - April 1 – 10% probability, July 1 – 50%, etc.

Other Estimation Factors

- Account for resource experience or skill
  - Up to a point
  - Often needed more on the “low” end, such as for a new or junior person
- Allow for “non-project” time & common tasks
  - Meetings, phone calls, web surfing, sick days
- There are commercial ‘estimation tools’ available
  - They typically require configuration based on past data

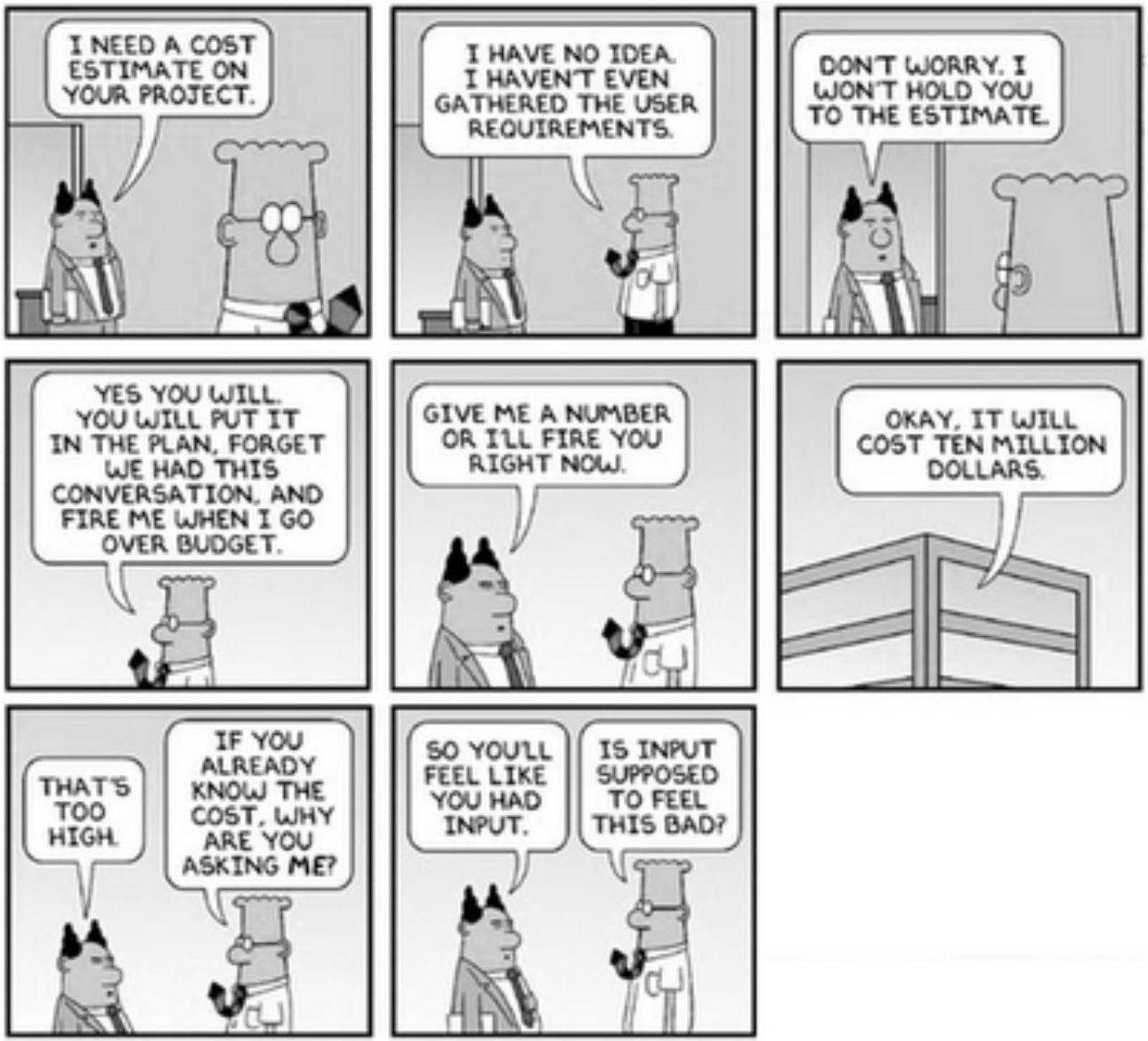
Other Estimation Notes

- Remember: “manage expectations”
- Parkinson’s Law
  - “Work expands to fill the time available”
- The Student Syndrome
  - Procrastination until the last minute (cram)

Optional Reading

- McConnell: 9, “Scheduling”
- Schwalbe: 5, “Project Time Management”

Comix


I NEED A COST ESTIMATE ON YOUR PROJECT.
I HAVE NO IDEA. I HAVEN'T EVEN GATHERED THE USER REQUIREMENTS.
DON'T WORRY. I WON'T HOLD YOU TO THE ESTIMATE.
YES YOU WILL. YOU WILL PUT IT IN THE PLAN, FORGET WE HAD THIS CONVERSATION, AND FIRE ME WHEN I GO OVER BUDGET.
GIVE ME A NUMBER OR I'LL FIRE YOU RIGHT NOW.
OKAY, IT WILL COST TEN MILLION DOLLARS.
THAT'S TOO HIGH.
IF YOU ALREADY KNOW THE COST, WHY ARE YOU ASKING ME?
SO YOU'LL FEEL LIKE YOU HAD INPUT.
IS INPUT SUPPOSED TO FEEL THIS BAD?

Questions?